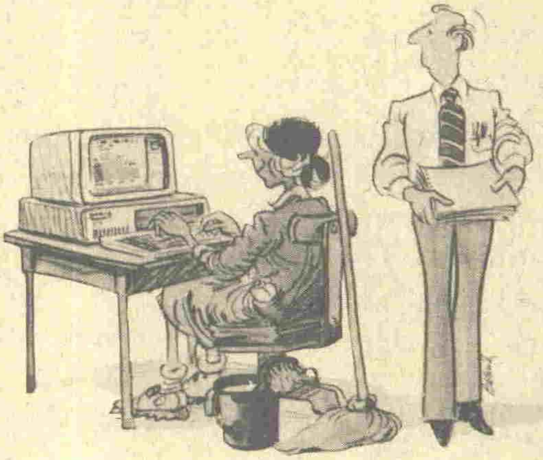
จเร เลิศสุดวิชัย

SOFT MAIL

พัฒนาโปรแกรมยูทิลิตี้ด้วย เทอร์โบซี

ตอนที่ 6

นำเสนอโปรแกรมยูทิลิตี้ที่ใช้รักษาความปลอดภัยข้อมูล และช่วยขจัดไฟล์ที่เป็นไฟล์แบคอัพ รวม 2 โปรแกรม



ในมุมภาษาซีฉบับนี้ประกอบด้วยสองโปรแกรมด้วยกัน โดยโปรแกรมแรกคือโปรแกรม WF (Wipe File) ซึ่งเป็นโปรแกรมที่เกี่ยวกับการรักษาความปลอดภัยของข้อมูล โดยจะช่วยให้การลบไฟล์ข้อมูลนั้นเป็นการลบข้อมูลออกจากดิสค์จริง ๆ ไฟล์ที่ถูกลบด้วย WF จะไม่สามารถกู้กลับคืนมาใหม่ได้อีก ถึงแม้จะใช้โปรแกรม Undelete หรือโปรแกรมยูทิลิตี้อื่น ๆ ก็ตาม โดยโปรแกรม WF จะทำให้ข้อมูลในไฟล์เป็นศูนย์เสียก่อนที่จะมีการลบไฟล์จริง

ส่วนโปรแกรมที่สองคือโปรแกรม CB (Clean BAK) ซึ่งเป็นโปรแกรมที่ช่วยขจัดไฟล์ที่เป็นไฟล์แบคอัพ (BAK) ที่เกิดจากการแก้ไขไฟล์ต่าง ๆ โดยตามปกติแล้วไฟล์ .BAK เหล่านี้มักจะไม่ได้รับความสนใจและถูกปล่อยทิ้งไว้ ซึ่งหากมีจำนวนมากขึ้นจะเสียพื้นที่ดิสค์โดยเปล่าประโยชน์

โปรแกรม WF (Wipe File)

การลบไฟล์ออกจากดิสค์ด้วยคำสั่ง del ของดอสนี้ยังสามารถที่จะกู้คืนกลับมาได้ โดยอาจจะใช้คำสั่ง undelete ของดอส (รุ่น 5.0 เป็นต้นไป) หรืออาจใช้โปรแกรมยูทิลิตี้อื่น ๆ เช่น นอร์ตันยูทิลิตี้ พีซีทูล เป็นต้น ซึ่งเป็นจุดหนึ่งที่ต้องระมัดระวังในเรื่องความปลอดภัยข้อมูล สาเหตุที่สามารถกู้ไฟล์คืนกลับมาได้อีกเมื่อถูกลบด้วยคำสั่ง del นั้น เป็นเพราะว่าการลบไฟล์ที่เกิดจากคำสั่งนี้ไม่ได้มีการลบข้อมูลเนื้อไฟล์ออกจากดิสค์จริง แต่ยังคงปล่อยข้อมูลทิ้งไว้อยู่ ซึ่งหากเนื้อที่ดิสค์ในบริเวณนั้นยังไม่ได้ถูกนำไปใช้งานอย่างอื่น การกู้ไฟล์กลับคืนมาก็สามารถทำได้อย่างง่ายดาย

ดังนั้นเพื่อป้องกันความลับของข้อมูลในไฟล์ เมื่อลบออกจากดิสค์ควรจะถูกลบออกอย่างสมบูรณ์ วิธีหนึ่งที่พอจะทำได้คือการเขียนข้อมูลอื่นทับลงไปในเนื้อข้อมูลของไฟล์นั้นก่อนจะทำการลบทิ้ง ซึ่งโปรแกรม WF จะสามารถช่วยงานทางด้านนี้ได้ โดยการทำงานของโปรแกรม WF นั้นจะลบไฟล์ออกจริงโดยจะทำการเขียนข้อมูลแอสกีศูนย์ลงแทนในข้อมูลของไฟล์ก่อนที่จะมีการลบไฟล์ออกจริง นอกจากนั้นยังทำการเปลี่ยนวันเวลา วันที่ และขนาดของไฟล์ให้มีค่าเป็นศูนย์ด้วย รวมทั้งจะมีการเปลี่ยนชื่อไฟล์เดิมให้เหลือเพียงตัวอักษรตัวเดียว เพื่อเป็นการป้องกันอีกชั้นหนึ่ง ซึ่งทำให้การติดตามและตรวจหาชื่อไฟล์เพื่อจะกู้คืนยากขึ้นอีกขั้นหนึ่ง

รูปแบบการใช้งาน

```
c: \ >wf filename.ext
```

โดยที่ filename.ext เป็นชื่อไฟล์ที่ต้องการจะลบออกจากดิสค์ ซึ่งการระบุชื่อไฟล์สามารถระบุเป็นกลุ่มไฟล์ได้ด้วยตัวอักษร Wildcard นอกนั้นยังสามารถระบุชื่อไฟล์และไดเรคทอรีของไฟล์ที่ต้องการลบไว้หน้าไฟล์ได้ด้วย

ตัวอย่างที่ 1

```
A: \ >wf customer.WK1
```

เป็นการลบไฟล์ customer.WK1 ซึ่งเป็นไฟล์ของโปรแกรมโลตัส

ตัวอย่างที่ 2

```
C: \ >wf D: \ msword \ doc \ secret.doc
```

เป็นการลบไฟล์ชื่อ secret.doc ซึ่งอยู่ไดรฟ์ d และในไดเรคทอรี \ msword \ doc

ตัวอย่างที่ 3

```
C: \ >wf note.wri
WF – Wipe File, TC Edition
File dose not exist
```

หากชื่อไฟล์ที่ระบุให้ลบไม่มีอยู่จริง โปรแกรม wf จะรายงานผลมาให้ทราบว่าไม่สามารถค้นหาชื่อไฟล์ที่ระบุได้

ตัวอย่างที่ 4

```
A: \ >wf
```

หากเรียก wf เพียงอย่างเดียว โปรแกรมจะแสดงวิธีการใช้งานโปรแกรมให้ดู

เทคนิคโปรแกรม

การทำงานของโปรแกรม wf ไม่ซับซ้อนอะไรมากนัก โดยจากฟังก์ชัน main หลังจากได้รับชื่อไฟล์ที่ระบุให้ทำการลบมาแล้วก็จะแยกชื่อไฟล์ออกเป็นส่วนต่าง ๆ คือ ชื่อไดรฟ์ ชื่อไดเรคทอรี และชื่อไฟล์จริง ๆ เอง (เพื่อที่จะนำไปใช้งานในการเปลี่ยนชื่อในภายหลัง) จากนั้นจะเปิดไฟล์แล้วเขียนแอสกีศูนย์ลงไปในไฟล์ ซึ่งเทคนิคการเขียนข้อมูลนั้นเขียนทีละ 32 กิโลไบต์ ทั้งนี้เพื่อให้โปรแกรมทำงานได้เร็วขึ้น ดังนั้นจะเห็นว่าในฟังก์ชัน wipefile จะมีการจองหน่วยความจำขนาด 32 กิโลไบต์เพื่อใช้เป็นบัฟเฟอร์ในการเขียนข้อมูลลงไฟล์ และหลังจากนั้นจึงปิดไฟล์ก่อนแล้วเปิดไฟล์ใหม่ แต่ครั้งนี้จะเปิดไฟล์เพื่อเขียนหรือเท่ากับเป็นการสร้างไฟล์ใหม่ ซึ่งจะเป็นผลทำให้ขนาดของไฟล์มีค่าเป็นศูนย์ทันที การเปิดไฟล์ครั้งที่ 2 นี้จะใช้ฟังก์ชัน fopen (filestr, "w") ซึ่งต่างจากการเปิดไฟล์ครั้งแรกที่ใช้ฟังก์ชัน fopen (filestr, "r+") ซึ่งเป็นการเปิดไฟล์เพื่ออ่านและเปลี่ยนแปลงแก้ไข

จากนั้นจะเริ่มเปลี่ยนเวลา วัน เดือน ปีของไฟล์ให้เป็นศูนย์แล้วจึงปิดไฟล์อีกครั้ง และก่อนที่จะมีการลบทิ้งจริงด้วยฟังก์ชัน unlink ( ) โปรแกรมจะทำการเปลี่ยนชื่อไฟล์เป็นตัวอักษรรหัส 255 ตัวอักษรเดียว ทั้งนี้ก็เพื่อเป็นการป้องกันและสร้างความลำบากให้กับผู้ที่คิดจะกู้ไฟล์คืนจะไม่สามารถหาชื่อไฟล์เดิมได้

โปรแกรม CB (Clean Blank)

โปรแกรม CB เป็นโปรแกรมที่ช่วยจัดการทำความสะอาดดิสค์เก็บข้อมูลให้กับระบบ โดยโปรแกรมจะทำการค้นหาไฟล์ที่มีนามสกุล .BAK ในทุกไดรฟ์ช่องเก็บข้อมูล โดยเฉพาะฮาร์ดดิสค์และไดรฟ์เน็ตเวอร์ก เนื่องจากไฟล์นามสกุล .BAK คือไฟล์แบคอัพซึ่งเกิดขึ้นเสมอเมื่อมีการแก้ไขเปลี่ยนข้อมูลในไฟล์ต่าง ๆ ซึ่งผู้ใช้โดยส่วนใหญ่มักจะไม่ค่อยให้ความสนใจไฟล์เหล่านี้นัก และไฟล์เหล่านี้จะเพิ่มจำนวนมากขึ้นตามเวลาของการทำงานซึ่งหากเก็บไว้ต่อไปก็จะกินพื้นที่ดิสค์โดยไม่มีประโยชน์

โปรแกรม CB จะมีการทำงานเช่นเดียวกับโปรแกรม ff (Find File) ซึ่งได้เคยนำเสนอไปแล้วในฉบับที่ 96 แตกต่างกันก็เพียงแต่ว่าโปรแกรม CB จะค้นหาเฉพาะไฟล์ที่มีนามสกุล .BAK เท่านั้น และเมื่อค้นพบจะทำการลบไฟล์เหล่านั้นทิ้งทันที

รูปแบบการใช้งาน

```
C: \ >cb
```

การใช้งานโปรแกรม cb นั้นเพียงเรียกชื่อโปรแกรมเท่านั้น ไม่ต้องระบุอย่างอื่น โปรแกรมจะทำการตรวจสอบเองว่ามีฮาร์ดดิสค์ติดตั้งอยู่ในระบบกี่ตัว และดิสค์ที่เน็ตเวอร์กมีกี่ตัว ซึ่งเป็นเทคนิคเดียวกันกับโปรแกรม ff นั่นเอง

จากนั้นโปรแกรมจะเริ่มต้นที่ฮาร์ดดิสค์ตัวแรกแล้วค้นไฟล์นามสกุล .BAK ในทุก ๆ ไดเรคทอรีของดิสค์ตัวนั้น ถ้าพบไฟล์ในนามสกุล .BAK จะลบทิ้งทันที และทำเช่นนี้กับทุกไดรฟ์ในระบบและกับทุก ๆ ไดเรคทอรีย่อย

เทคนิคโปรแกรม

ในโปรแกรม cb มีฟังก์ชันที่สำคัญมากและก็ใช้งานอยู่ในโปรแกรม ff ด้วยเช่นกัน ซึ่งก็คือฟังก์ชัน getdirectory ( ) ซึ่งจะสร้างเส้นทาง (path) ของไดเรคทอรีปัจจุบัน เส้นทางดังกล่าวจะประกอบไปด้วยชื่อไดรฟ์และชื่อไดเรคทอรีรวมกัน เพราะถ้ามีแต่เพียงการทำงานของฟังก์ชัน getcurdir ( ) เท่านั้น จะได้เพียงชื่อไดเรคทอรีเท่านั้น ซึ่งยังไม่สามารถนำเอาไปใช้งานได้ ต้องรวมกับชื่อไดรฟ์ซึ่งจะได้จากฟังก์ชัน getdisk ( ) นั่นเอง ดังจะเห็นได้จากการทำงานในฟังก์ชัน getdirectory ( )

จะเห็นได้ว่าโครงสร้างและการทำงานของโปรแกรม cb จะเป็นเช่นเดียวกันกับโปรแกรม ff เกือบทั้งหมด ซึ่งมุมภาษาซีได้เคยอธิบายไปบ้างแล้ว จึงจะไม่ขออธิบายซ้ำอีก

ต้นฉบับของโปรแกรม

ต้นฉบับของโปรแกรม wf.c และ cb.c พิมพ์อยู่ในรูปที่ 1 และ 2 และเช่นเดิมครับ สำหรับท่านผู้อ่านที่ไม่ต้องการพิมพ์เองก็จะหาได้จากซอฟต์เมล์ไมโครฯ ฉบับนี้ครับ

รูปที่ 1 โปรแกรม wf.c

```
/*

    Title          :  WF.EXE  (Wipe File)
    Purpose        :  This utility let you completely wipe all traces of
                       a file. It will overwrite a file with zeroes and
                       then also overwrite the date, time and file size
                       with zeroes.  Before deleting what is left, the
                       utility will also change the file name to a one
                       character name that protect it from undeleting them.
    requirements   :  The program will work on MS-DOS and PC-DOS computer
                       using version 2.0 or later

*/

#include <stdio.h>
#include <string.h>
#include <alloc.h>
#include <dos.h>
#include <dir.h>
#include <io.h>

/*
Only internal procedure used in this program. It return an
integer which is used for setting the DOS errorlevel cariable.
*/

int wipefile (void);

/*
Global variables used throughtout this program.
*/

char filestr [MAXPATH];                /* User input */
char drive  [MAXDRIVE], subdir [MAXDIR],
    file [MAXFILE], ext [MAXEXT],
    fullname [MAXPATH];                /* Use by spliting function */

char newname [2];                      /* Fake name is stored here */

int problem;                           /* Holds return value */

/*
The main loop of this utility obtains arguments from the command line.
If the command line does not have enought arguments, a help
 message is
printed and the program ends with an error code. If everything is alright,

then string variable FILESTR is set to be equal to the first parameter and

a NEWNAME string variable is set equal to the ASCII 255 that can't entered

from the DOS.

CALLED FROM:   (DOS)

CALLS:        wipefile

*/

main (int argc, char *argv[])
{
  printf ("\nWF-Wipe File, TC Edition.\n");
  if (argc > 1)
                  {
                  strcpy (filestr,argv[1]);
                  fnsplit (filestr, drive, subdir, file ,ext);
                  sprintf (fullname,"%s%s", drive,subdir);
```

→

รูปที่ 1 โปรแกรม wf.c (ต่อ)

```
                }
        else
                {
                printf ("\nUsage: C:\>wp filename.ext\n");
                printf ("\nfilename.ext is the name of file you want to delete.\n");
                return (1);
                }
        newname [0] = 255;
        newname [1] = 0;
        strcat (fullname, newname);
        problem = wipefile ();

        if (problem)
                return (1);
        else
                return (0);
}

/*
This procedure allocates a buffer set to zeroes and writes it out over
the file as many time as necessary to overwrite the entire file. The
file is then closed and opend again for writing. Because of this, the
file size goes to zeroes. Then the file's date and time is also set to
zero with a Turbo C function call and the file is rename and deleted.

CALLED FROM:  main
CALLS:        (none)

*/


int wipefile (void)
{
        char    *buffer;
        int     handle, buffersize;
        unsigned long size;
        long    filepos;
        FILE    *oblitfile;
        struct  ftime filetime;

        buffer = (unsigned char *) calloc (32767,1);

        if (buffer == NULL)           /* Not enough memory. */
                return (1);

        oblitfile = fopen (filestr, "r+"); /* Open file for update. */
        if (oblitfile == NULL)
                {
                printf ("\nFile does not exist.\n");
                return (1);
                }
        fseek (oblitfile, 0, 0);       /* Start at the beggining of file */
        handle = fileno (oblitfile);

        size = ftell (oblitfile);

        if (size <= 32767)
                buffersize = size;
        else
                buffersize = 32767;
```

→

```
        filepos = 0;

        do
                {
                fseek (oblitfile, filepos, 0);
                _write (handle, buffer, buffersize);
                filepos = filepos + buffersize;

                if ((filepos + buffersize) > size)
                  buffersize = size - filepos;

                }
        while (filepos < size);

        fclose (oblitfile);
        free (buffer);

        oblitfile = fopen (filestr,"w");     /* Open for writing */
        handle   = fileno (oblitfile);      /* size is now zero */

        filetime.ft_tsec               = 0;
        filetime.ft_min                = 0;
        filetime.ft_hour               = 0;
        filetime.ft_day                = 0;
        filetime.ft_month  = 0;
        filetime.ft_year               = 0;
        setftime (handle, &filetime);        /* Set the time to zeroes */

        fclose (oblitfile);

        rename (filestr, fullname);          /* Rename the file to 255 */
        unlink (fullname);                  /* and erase it. */
        return (0);

}
```

■

รูปที่ 2 โปรแกรม cb.c

```
/*

  Title           :   CB.EXE    (Clean Blank)
  Purpose         :   This utility will allow the quick deletion of all
                      files ending in "BAK". CB rids your hard disk and
                      or network of useless backup files by systematically
                      checking each drive and directory for BAK files and
                      deleting them.
  requirements    :   The program will work on MS-DOS and PC-DOS computer
                      using version 2.0 or later

*/

#include <stdio.h>
#include <string.h>
#include <dir.h>
#include <dos.h>

/*
The two internal procedures used in this program. The GETDIRECTORY
function returns a pointer to a text string.
*/
```

→

รูปที่ 2 โปรแกรม cb.c (ต่อ)

```
void findfile (void);
char *getdirectory (char *path);

/*
No global variables are used by this utility.
*/

/*
The main loop of this utility does several interesting things. One
of these is the creation of a table of valid drives. It assumes that
a system can have 26 drives (one per letter of the alphabet) and procedes
to check the amount of space on each drive. If it gets a valid number back
from the Turbo C function, it assumes that a drive exists. The scanning
for drives is started with the C drive.

CALLED FROM:   (DOS)
CALLS:         findfile     getdirectory

*/

main ()
{
        /* Variables used in the main program. */

        char origdir [MAXPATH], rootdir [MAXPATH],
           tempdir [MAXPATH], availdrvs [26];
        int i;
          struct dfree freespace;

          printf ("\nCB-Clean BAKs, TC Edition.\n");

          for (i = 3; i < 26; i++)         /* Test for the existence of drives */
            {
                getdfree (i, &freespace);
                if (freespace.df_sclus == 0xFF) /* Character 255 */
                        availdrvs [i] = 0;         /* No drive */
                else
                        availdrvs [i] = 1;
                }

          getdirectory (origdir);          /* Get current directory */

          for (i = 3; i <= 26; i++)
                {
                if (availdrvs [i] == 1)
                        {
                        rootdir [0] = '@' + i;
                        rootdir [1] = ':';
                        rootdir [2] = 92;
                        rootdir [3] = 0;
                        setdisk (i -1 );
                        getdirectory (tempdir);
                        chdir (rootdir);           /* Switch to root */
                        findfile ();
                        chdir (tempdir);
                        }
                }
          setdisk (origdir [0] - 65);      /* Back to original */
          chdir (origdir);
          return (0);
}

/*
This procedure searchs for files that match the "*.BAK". Once a
match is found, it is deleted with the Turbo C function UNLINK. Each
directory oof each drive is scanned for backup files. This procedure
is recursive.

CALLED FROM:   main
CALLS:         getdirectory

*/

void findfile (void)
{
          /*
          Variables used in the procedure.
          */

          char curdir [MAXPATH], temp [3];
          int  done, attrib;
          struct ffblk dta;

          getdirectory (curdir);
          attrib = FA_RDONLY + FA_HIDDEN + FA_SYSTEM;
                                          /* All files minus directory */
          done = findfirst ("*.BAK", &dta, attrib);
          if (!done)
                  printf ("\nCleaning --> %s\n\n",curdir);

   while (!done)
                  {
                  unlink (dta.ff_name);       /* Delete all found .BAK files. */
                  done = findnext (&dta);
                  }

                  attrib = FA_DIREC + FA_HIDDEN + FA_SYSTEM;
                  done = findfirst ("*.*", &dta, attrib);
                  while (!done)
                    {
                    strncpy (temp, dta.ff_name, 2);
                    if (((dta.ff_attrib & FA_DIREC) == FA_DIREC) && (temp[0] != '.'))
                            {
                            if (strlen (curdir) != 3)
                              strcat (curdir,"\\");

                            strcat (curdir, dta.ff_name);
                            chdir(curdir);
                            findfile();
                            chdir ("..");
                            getdirectory (curdir);
                            }
                    done = findnext (&dta);
                    }
}

/*
This procedure creates a string which holds the druve and path where
the program is searching for backup files. The Turbo C function GETDISK
is called to get the disk drive number. This number is added to the
letter A (the first diskdrive) tro get the letter of the current disk
drive. When the procedure is over, a pointer to drive and path is
return to the caller

CALLED FROM:   main       findfile
CALLS:                    (none)
*/

char *getdirectory (char *path)
{
  strcpy  (path, "X:\\");
  path[0] = 'A' + getdisk ();
  getcurdir (0, path + 3);        /* Place it in path area of
                                              (3rd position on) the string */
  return (path);
}
```